面向高速网络内容安全处理的串匹配算法[1]

何慧敏 刘夏 姜磊

摘要：互联网技术的普及和发展使得网络安全形势日趋严重，对网络内容安全处理技术提出了更高的要求，而作为网络内容安全处理核心技术的模式串匹配算法也面临着新的挑战。本文针对高速网络内容安全处理的需求，提出了三种类型的串匹配算法，包括软件算法 (如：最优窗口选择算法)；指令集算法(如：基于 SSE 指令集的串匹配算法)和硬件算法(如：基于 FPGA 的串匹配算法)。它们极大地提高了模式串匹配的速度，符合高速网络内容安全处理系统的要求。

关键词：高速网络内容安全处理；最优窗口选择；指令集算法；硬件算法

1 背景与研究意义

近年来，随着互联网技术的普及和发展，互联网协议和计算机系统的漏洞造成的网络安全形势日趋严峻。传统的网络内容安全处理技术的存储空间和运算速度已经难以满足实时高速网络环境下的应用需求，成为限制网络内容安全处理技术发展的瓶颈。而面向高速网络内容安全处理的串匹配技术处理速度快、实时性高，已经得到学术界和工业界的广泛关注，并成为网络安全领域的研究热点。

模式串匹配算法是网络内容安全处理的核心技术，也是计算机科学领域研究的焦点之一。在网络安全领域，网络内容安全处理技术主要采用模式串匹配算法（Pattern Matching Algorithms）来分析网络包的内容负载，并根据预先设定的规则对特定的网络数据进行识别和分类。其中，入侵检测/防御系统（IDS/IPS）、反病毒和反垃圾邮件检测（AV/AS）、网络带宽管理和服务质量（QoS）、统一威胁管理（UTM）都是较为典型的应用。

所谓模式串匹配是指：给定模式串集合 ***P***，对于任意的输入文本 ***T***，找出 ***P*** 中的模式串在 ***T*** 中的所有出现位置。广义的模式串匹配算法包括：精确多模式串匹配算法、正则表达式匹配算法和近似串匹配算法。精确多模式串匹配是串匹配技术中最成熟的一种技术，已经在入侵检测系统中得到广泛的应用，并且成为影响系统性能的决定性因素。而近年来，构成网络内容安全威胁的病毒规则日益复杂，在应用系统中，有时精确串规则不能精确地表达用户的语义需求，这就需要使用正则表达式表达更复杂的语义。本文的工作从精确串匹配和正则表达式匹配两方面开展。

2 相关工作

模式串匹配算法是网络内容安全处理的核心技术，也是关系到文本检索、搜索引擎、语言翻译、拼写检查、生物计算等领域发展的关键因素，因此近年来出现了各种对经典串匹配算法的改进工作，新的算法设计也层出不穷。

近年来，国内外研究者在精确串匹配算法研究方面产生了很多新的结果。文献[1]提出

[1] 本文研究工作得到国家自然科学基金项目“面向高速网络内容安全处理的专用系统结构”（项目批准号：61070026）支持

了一种时间复杂度最优的精确串匹配算法 LDM（Linear DAWG Matching，线性 DAWG[2]匹配）。该算法将文本划分为相互重叠的窗口，并在每个窗口内综合使用后缀自动机 DAWG 和 AC[3]自动机进行扫描，保证了线性的最坏时间复杂度和亚线性的平均时间复杂度；文献[2，3]使用布隆过滤器（Bloom Filter）技术进行串匹配，将特征串按照长度进行分组，相同长度的特征串被分在同一组，每一组用一个布隆过滤引擎进行过滤。

学术界对正则表达式匹配技术也进行了深入研究。文献[4]用规则分类和规则改写的方法来化简正则表达式，文中提出了两条改写规则以降低线性规模的状态数，但是这仅适合于非重叠匹配的情况，因此限制了它的使用范围；文献[5，6]提出了输入延时有限自动机（Delayed input Deterministic Finite Automata， D2FA）和 delta-FA（Delta Finite Automata）方法来压缩确定自动机的存储空间。它们都是利用相邻状态之间的关系来减少转换表中转移的数目，从而降低自动机的存储空间，但是都有状态更新费时的缺陷，所以实际匹配性能不高。

尽管现有的串匹配算法众多，但仍存在以下几点不足：(1)这些算法只能处理中等规模以下的模式串集合，当模式串集合规模扩大时，处理性能大幅度降低；(2)这些算法只对某些特定类型的模式串具有一定的效果，不具有普遍性；(3)上述硬件算法在实际系统中并不能满足实时处理网络内容的要求。因而，仍需要我们设计更好的模式串匹配算法。

3　我们的研究工作介绍

我们从精确串匹配技术和正则表达式匹配技术入手，提出软件算法、指令集算法和硬件算法三种类型的串匹配算法。其中，软件算法设计了最优窗口选择算法；指令集算法利用 SSE 指令集对 SOG 算法和经典的位并行算法优化；硬件算法实现了基于现场可编程门阵列（FPGA）的正则表达式匹配算法。这些算法在提高匹配速度方面取得了良好效果。

3.1 软件算法：最优窗口选择算法

在串匹配算法中，后缀算法因其高效的匹配速度而得到广泛应用，但它要求模式串等长，这个条件在实际应用中往往无法满足。一般的后缀算法处理方法非常简单，如吴--曼博（Wu-Manber）[7]算法选取模式串最左边的长度为 ***m*** 的部分串，***m*** 为模式串中最短串的长度。这种处理方法没有考虑选取的部分串对匹配性能的影响。我们提出一种子串提取方法，从模式串集合中提取出部分串集合构造基本数据结构，用此数据结构扫描文本串，以达到尽可能大的匹配速度。

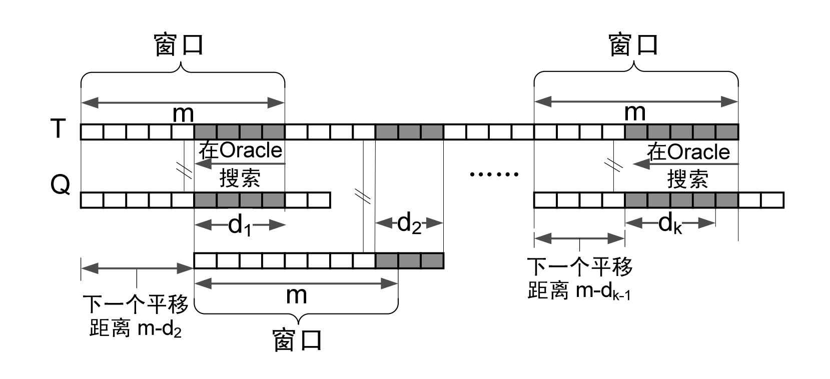


图1.　SBOM 算法找到所有模式串出现的位置

我们采用 SBOM(Set Backward Oracle Matching)[8]算法在训练过程完成模式串集合的部

[2] Directed Acyclic Word Graph，有向无环字图

[3] Aho-Corasick，阿霍-克若思克

分串集合的提取，见图 1。用 T 表示文本串，n 表示文本的长度，m 表示模式串集合 $P=\{p_1,p_2,\cdots,p_r\}$ 中最短的模式串长度，k 表示搜索文本串时滑动窗口的总次数，d_i 表示第 i 个窗口两个对应字符的比较次数，则总的时间开销为 $\sum_{i=1}^{k} d_i$，k 满足等式 $km-\sum_{i=1}^{k} d_i = n$，则可得公式 1 和公式 2：

$$totalcost = km - n \quad (1)$$

$$averagecost = \frac{km}{n} - 1 \quad (2)$$

由公式 2 可知，k 最小时，平均开销最小。而 k 由长度为 m 的部分串集合 $Q=\{q_{1i},q_{2j},q_{3k}\cdots,q_{rh}\}$ 决定，但由于 Q 的数目过大，因此无法通过遍历 Q 得到最小的 k 值。故我们采取一种近似最优的方法，对一条模式串的每个部分串计算其时间开销，从而可选出时间开销最小的部分串，将所有部分串并起来即可得到 Q。

具体步骤如下，对模式串 p_i 中的任一部分串 q_{ij}，

步骤 1：首先计算其所有子串 $s_{j,k}^i$，然后计算每个子串在训练文本中的出现次数 $c_{j,k}^i$，设 $s_{j,k}^i$ 的长度为 z， 则每个子串的时间开销为 $c_{j,k}^i \times z$；

步骤 2：将所有子串的时间开销累加，计算得到每个部分串的时间开销；

步骤 3：重复步骤 1 和步骤 2 计算该模式串中其它部分串的时间开销，然后选出时间开销最小的部分串；

步骤 4：重复步骤 1、步骤 2 和步骤 3 计算其它模式串的最佳部分串。

我们提取 Snort[9]中的模式串为实验数据，可以得到图 2 所示的结果。在图 2 中，“right”、“left”和“middle”分别指的是从最右、最左和中间三个位置提取 m 个字符。Mincost 代表我们算法提取的子串。从图中可以看出，虽然四种算法随着模式串规模的增长，匹配速度都呈现下降趋势。但是我们的算法同其它三种子串提取的方法相比，速度更快，比 right 方式快 20%左右，并且随着模式串规模的增长，我们算法的速度下降更慢。

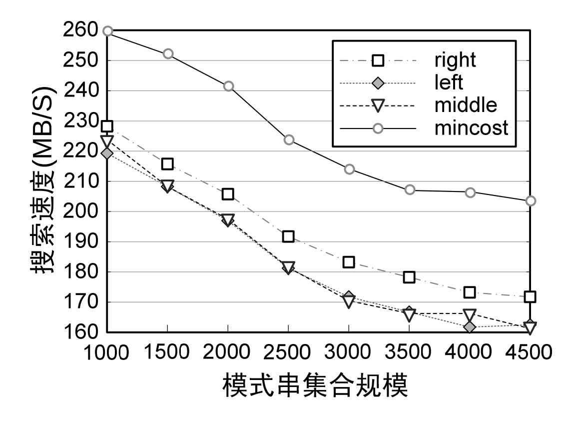


图2. 从不同位置提取部分串的匹配速度与模式串规模变化关系

3.2 指令集算法：基于 SSE 指令集的串匹配算法

SSE（Streaming SIMD Extensions 单指令多数据流扩展）系列指令集提供了一组大位宽的寄存器，支持 128 位数据的并行操作。本文利用 SSE 指令集对经典的位并行算法（Shift-And 和 BNDM）和 SOG 算法进行优化。

Shift-And 算法和 BNDM 算法是经典的位并行算法，设机器字的长度为 w，文本串的长度为 n，模式串的个数为 r，最短模式串长度为 m，那么 Shift-And 算法和 BNDM 算法的时间复杂度为 $O(n\lceil mr/w \rceil)$。由于采用了位并行技术，Shift-And 算法和 BNDM 算法的匹配速度是很快的。但一旦模式串的长度和超出机器字的长度，两种算法的性能都会发生明显下

降。

利用 SSE 指令集对 Shift-And 算法和 BNDM 算法进行优化的基本思想是：采用位向量 $\boldsymbol{D}$ 记录每个模式串前缀或子串和当前文本 $\boldsymbol{t}$ 的匹配状态，将多个模式串的状态向量打包（pack）到 128 位的 SSE 寄存器上（如图 3 所示）。在匹配的过程中，每读入一个文本串字符就通过 SSE 指令集提供的位操作指令（逻辑“或”、逻辑“与”、左移）更新状态向量 $\boldsymbol{D}$，并通过 SSE 指令集提供的位比较指令来检测特定的位以判断模式串和文本是否匹配成功。

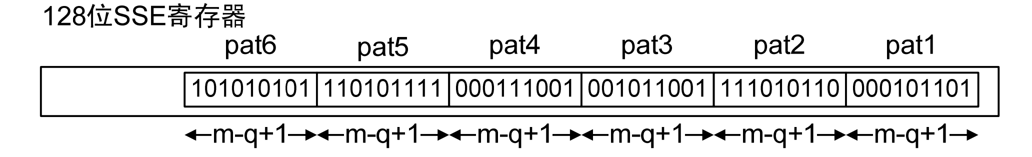


图3.　Shift-And 算法、BNDM 算法和 SOGOPT 算法过滤阶段的 SSE 指令优化示意图

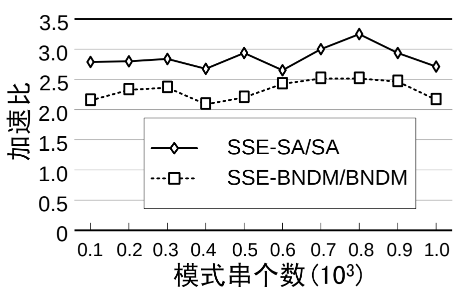


图4.　优化算法和原始算法的速度比值（模式串长度为 6B）

实验中，把优化后的 Shift-And（称为 SSE-Shift-And）算法和优化后的 BNDM（SSE-BNDM）算法与原算法比较（如图 4），可知 SSE-Shift-And 算法和 SSE-BNDM 算法都达到了良好的加速效果，且随着模式串个数的增加，两种算法的加速比基本保持不变。

SOG 算法[10]是 L. Salmela 等人提出的面向 10 万规模模式串的匹配方法。该算法将模式串集合

$$\boldsymbol{P}=\left\{\boldsymbol{p}^{(1)},\boldsymbol{p}^{(2)},\cdots,\boldsymbol{p}^{(r)}\right\}$$

规约为一个通配形式的模式串 $\boldsymbol{p}$；然后使用 Shift-OR 算法对 $\boldsymbol{p}$ 进行搜索；最后使用二级哈希和二分搜索校验可能出现的位置，并报告出所有真正匹配的结果。由 SOG 算法基本原理可知其过滤阶段的误报率为

$$P_f=(r/\sigma^{q})^{m-q+1}(1-1/(r^{m-q}))\text{。}$$

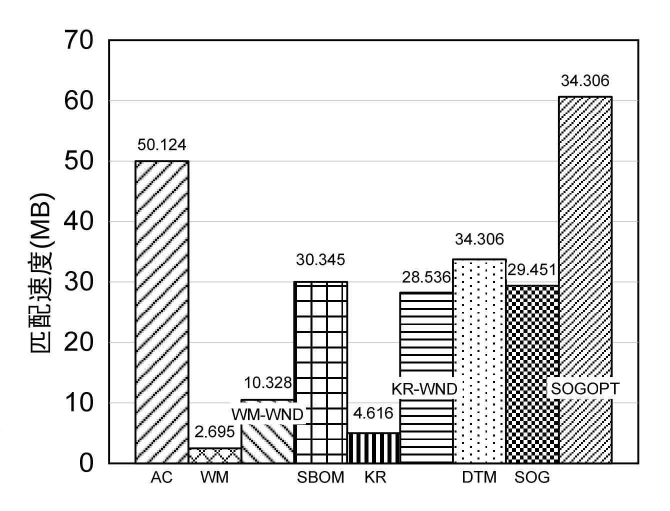


图5.　算法匹配速度对比(17.5 万条 URL 模式串)

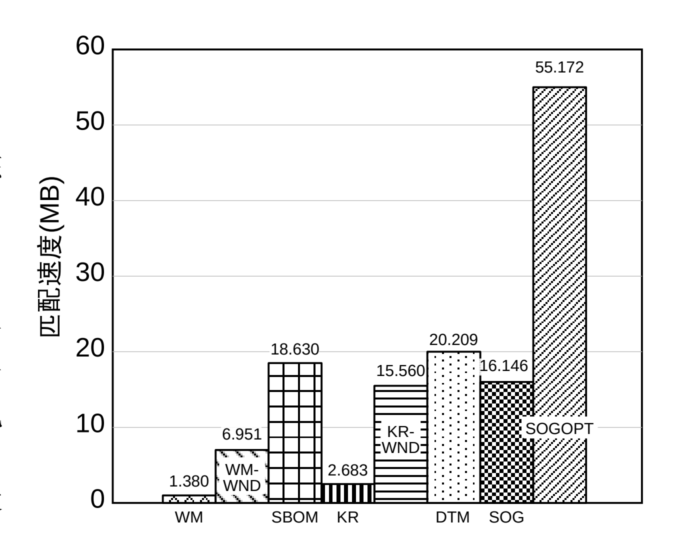


图6.　算法匹配速度对比(25 万条 URL 模式串)

随着模式串规模 $\boldsymbol{r}=|\boldsymbol{P}|$ 的增大，SOG 算法的过滤效果越来越差，需要进行校验的次数越来

越多，极大地降低了算法的匹配速度。为此，本文采用模式串分组归约的方法来改进 SOG 算法的过滤效果。

模式串分组规约基本思想是：将模式串集合 $\boldsymbol{P}$ 平均分为 $\boldsymbol{g}$ 组：$\boldsymbol{P}=\boldsymbol{P}_1\cup\boldsymbol{P}_2\cdots\cup\boldsymbol{P}_g$，将每一组模式串 $\boldsymbol{P_j}$ 单独规约为一个通配形式的模式串 $\boldsymbol{P}_j(1\le \boldsymbol{j}\le \boldsymbol{g})$，然后将这 $\boldsymbol{g}$ 个通配模式 $\boldsymbol{P_j}$ 拼接到一个机器字中，使用 SOG 算法进行并行搜索。设机器字位宽为 $\boldsymbol{w}$，因为每个通配模式 $\boldsymbol{P_j}$ 需要 $\boldsymbol{m}$-$\boldsymbol{q}$+1 比特位来表示，因此最多可以将模式串集合划分为 $\boldsymbol{g}=\lfloor \boldsymbol{w}/(\boldsymbol{m}-\boldsymbol{q}+1)\rfloor$ 组。

本文使用的算法是利用最大二部图匹配理论为模式串选择最优窗口，对 SOG 优化后形成的 SOGOPT 算法。我们在分组规约后的 SOGOPT 算法中引入 SSE 指令，利用 SSE 寄存器的位宽来减少算法的校验次数。具体方法是：采用位向量 $\boldsymbol{D}$ 记录每个通配符串的前缀与当前文本 $\boldsymbol{T}$ 的匹配状态，将状态向量打包到 128 位的 SSE 寄存器。在匹配过程中，每读入一个文本字符就通过 SSE 指令集提供的位操作指令更新状态向量 $\boldsymbol{D}$，并检测特定的位以判断是否匹配成功。

我们从骨干路由器上采集了约 100GB 的 URL[4]数据，从匹配速度和存储空间两个方面，将 SOGOPT 算法与原始的 SOG 算法以及多种串匹配算法 AC、WM、SBOM、KR、DTM 进行了对比测试。可以发现，SOGOPT 算法匹配速度最快，内存消耗也维持在合理范围之内。测试结果如图 5、图 6 所示。

3.3 硬件算法：基于现场可编程门阵列的串匹配算法

随着网络带宽的高速增长，正则表达式匹配的硬件方法备受关注，目前已经有多种基于非确定型有限自动机（Non-deterministic Finite Automaton，NFA）和确定型有限自动机（Deterministic Finite Automaton，DFA）的正则表达式硬件加速方案。其中非确定型有限自动机算法主要是基于逻辑电路实现，需要较多逻辑电路资源；而确定型有限自动机算法将状态转移表存放到内存中，需要较多内存资源。为了解决确定型有限自动机算法消耗内存资源多的缺点，同时使其兼具非确定型有限自动机高性能的优势，我们提出一种完全基于现场可编程门阵列逻辑电路的确定型有限自动机匹配改进算法。

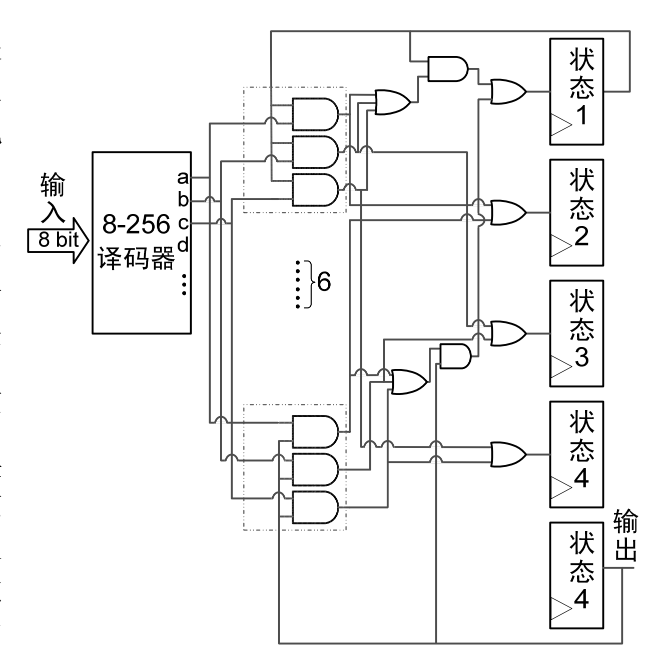


图7. “$(\boldsymbol{a}|\boldsymbol{b})^*\boldsymbol{cd}$”改进的 DFA 匹配电路

杨毅夫[11]指出确定型有限自动机中每个状态的大部分转移边会集中到相同的状态。我们可以合理利用确定型有限自动机的这个特征，将这些相同的转移边合并，称之为默认转移。默认转移指向的下一个状态为默认状态。而我们只需要记录这条默认转移边，就可以极大地减少转移边数，简化电路结构。

图 7 为正则表达式“$(\boldsymbol{a}|\boldsymbol{b})^*\boldsymbol{cd}$”改进后的确定型有限自动机的匹配电路。我们在电路中

[4] Uniform Resource Locator，统一资源定位符，即通常所说的网址

使用了一个与门和与非门的级联实现了默认转移边。每个状态转移表中存在的非默认状态的输出，除了连接到各自对应的状态寄存器之外，还要一起连接到一个由或非门和与门组成的默认转移电路上。如果当前状态激活，并且输入字符对应的下一个状态为默认状态时，该状态转移表中所有的与门输出都为低电平。此时或非门的输入变为高电平，表明了下一个状态应该是该状态对应的默认状态。使用这种方法，可以在状态转移表中减少大量不必要的逻辑门和电路的扇出系数，提高系统的吞吐率。对于正则表达式"$(a|b)^{*}cd$"来说，采用原始的确定型有限自动机方案需要 1543 个逻辑门，而改进之后逻辑门的数量下降到 38 个。

为了能够全面真实地比较改进后的确定型有限自动机匹配引擎的性能与斯德胡（R. Sidhu）[12]设计实现的非确定型有限自动机匹配引擎的性能，我们从 L7-filter 规则集中提取了 102 条正则表达式规则作为测试数据，分别记录了非确定型有限自动机算法和确定型有限自动机算法消耗的逻辑资源(LE)数量和时钟频率(MHz)。测试结果表明，有 62%的规则确定型有限自动机算法消耗的逻辑资源要少于非确定型有限自动机算法；有 10%左右的规则确定型有限自动机算法的吞吐率要高于非确定型有限自动机算法，60%的规则两种算法吞吐率持平，而有 30%的规则确定型有限自动机算法的吞吐率要低于非确定型有限自动机算法。

4　总结

本文设计了三种类型的串匹配算法，包括软件算法（最优窗口选择算法）、指令集算法（基于 SSE 指令集的串匹配算法）和硬件算法（基于现场可编程门阵列的串匹配算法），它们极大地提高了模式串匹配的速度，适用于高速网络内容安全处理系统。我们的下一步工作是设计适用于超大规模模式串的串匹配算法和面向高速网络内容安全处理的专用系统。

参考文献：

[1] 贺龙涛,方滨兴,于翔湛. 一种时间复杂度最优的精确串匹配算法. *软件学报*. 2005.5

[2] S. Dharmapurikar, P. Krishnamurthy, T. S. Sproull, J. W. Lockwood. Deep Packet Inspection using Parallel Bloom Filters. *IEEE Micro*, 24(1):52–61, 2004

[3] Michael Attig, Sarang Dharmapurikar, John Lockwood. Implementation Results of Bloom Filters for String Matching. *IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM), Napa, CA*, April 20-23, 2004

[4] Fang Yu, Zhifeng Chen, Yanlei Diao. Fast and memory-efficient regular expression matching for deep packet inspection. Proceedings of the 2006 ACM/IEEE symposium on Architecture for networking and communications systems 2006, San Jose, California, USA December 03-05, 2006

[5] Sailesh Kumar, Balakrishnan Chandrasekaran , Jonathan Turner , George Varghese, Curing regular expressions matching algorithms from insomnia, amnesia, and acalculia, Proceedings of the 3rd ACM/IEEE Symposium on Architecture for networking and communications systems, December 03-04, 2007, Orlando, Florida, USA.

[6] Domenico Ficara, Stefano Giordano, Gregorio Procissi, Fabio Vitucci, Gianni Antichi, Andrea Di Pietro, An improved DFA for fast regular expression matching, ACM SIGCOMM Computer Communication Review, Volume 38, Issue 5(October 2008),Pages 29-40

[7] S. Wu and U. Manber, "A fast algorithm for multi-pattern searching", Dept. of Computer Science, University of Arizona, Tucson, AZ, TR-94-17, 1994

（下转第 33 页）